# On distributed coordination in networks of cyber-physical systems

Cite as: Chaos **29**, 053126 (2019); https://doi.org/10.1063/1.5093728
Submitted: 24 February 2019 . Accepted: 07 May 2019 . Published Online: 31 May 2019

Giovanni Russo, and Mario di Bernardo

# On distributed coordination in networks of cyber-physical systems

Cite as: Chaos **29**, 053126 (2019); doi: 10.1063/1.5093728
Submitted: 24 February 2019 · Accepted: 7 May 2019 ·
Published Online: 31 May 2019



Giovanni Russo[1,a] and Mario di Bernardo[2,b),c)]

## AFFILIATIONS

[1]School of Electrical and Electronic Engineering, University College Dublin, Dublin 4, Ireland
[2]Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples 80125, Italy

**Note:** The paper is part of the Focus Issue "Complex Network Approaches to Cyber-Physical Systems".
[a)]**Electronic mail:** giovanni.russo1@ucd.ie
[b)]**Also at:** Department of Engineering Mathematics, University of Bristol, Bristol, United Kingdom.
[c)]**Electronic mail:** mario.dibernardo@unina.it

## ABSTRACT

This paper is concerned with the study of the global emerging behavior in complex networks where each node can be modeled as a cyber-physical system. We recast the problem of characterizing the behavior of such systems as a stability problem and give two technical results to assess this property. We then illustrate the effectiveness of our approach by considering two testbed examples arising in applications where networks, arising from Internet of Things applications, need to be designed so as to fulfill a given task.



**We live in a world where technological objects are becoming smaller, smarter, and with the ability of being interconnected. This trend is expected to continue at an extremely high pace, leading to systems where the physical world continuously interacts with a network of discrete technological entities. Systems arising from the "integration and interconnection" of continuous processes and procedural/algorithmic processes are known as cyber-physical systems[1–3] (CPSs). In this paper, we consider the problem of characterizing the emerging global behavior in networks where each node is a CPS, interacting with other nodes, and with the surrounding environment. After presenting two technical results to assess stability of such systems, we investigate the effectiveness of the approach by considering two testbed examples arising in the context of the Internet of Things.**

## I. INTRODUCTION

Over the past few years,[4–6] the study of interconnected nonlinear systems, or complex dynamical networks, attracted much research attention. In particular, a large body of literature has emerged, where the problem of studying the onset of collective, coordinated behaviors is considered. Synchronization[7] and consensus[8] are two remarkable instances of network coordination relevant to a number of applications as the study of opinion dynamics,[9] of bacterial quorum-sensing,[10] and of distributed generation in power grids.[11,12] The emergence of this coordinated behavior in a network can be studied in terms of convergence of the agents' states toward some synchronization manifold in state space.[13] Analogously, phenomena such as the formation of synchronized clusters of agents[14] in a network (or clustering) can be linked to agents' dynamics away from such a manifold.[15,16]

A notable example of a complex network is emerging through the "Internet of Things" (IoT) revolution[17] made possible by recent advances in computing and communication technologies. Essentially, IoT systems are complex networks, where each node can be linked physically (via sensors and actuators) to the surrounding environment and can be coupled to other nodes through a communication backbone. Systems arising from the integration and interconnection of continuous processes and procedural/algorithmic processes are known as cyber-physical systems (CPSs).

The problem considered in this paper is that of characterizing the global emerging behavior of networks, where each node is a CPS. One approach to address this problem is that of studying stability of these networks of CPSs. This approach is motivated by the fact that once stability of the system is proved, then the time evolution of the CPS can be qualitatively predicted, and, hence, the system can be designed so that it achieves a desired behavior. For example, results

can be given by modeling a CPS of interest as a hybrid system,[18,19] and control-theoretical tools[20,21] can then be used to assess stability. In the context of the above literature, the main contributions of this paper can be summarized as follows: (i) first, we present two technical results to characterize stability of a CPS of interest. Nonlinearities in the physical component can be considered, which interact with discrete/algorithmic processes; (ii) then, we show how the technical results can be used to design the physical and the cyber components of the system so as to guarantee the onset of some desired behavior; and (iii) finally, we illustrate the effectiveness of the approach to the design of network CPSs by considering two examples arising in the context of IoT. This paper is organized as follows. We start with formalizing, in Sec. II, the mathematical models considered in this paper. Two technical results to study stability of these systems are given in Sec. III. Then, in order to illustrate the effectiveness of our approach, in Sec. IV, we consider two testbed IoT applications. Finally, the mathematical tools and proofs of the results are given in Appendixes A and B.

## II. MATHEMATICAL MODEL

We consider CPSs of the form[22]

$$\begin{cases} \dot{x} = f(t,x,u), & x \in \mathbb{R}^n, \quad u \in \mathscr{U}, \\ u \leftarrow \mathscr{A}(x,e), & e \in \mathscr{E}, \quad x(t_0) = x_0, \quad t_0 \geq 0. \end{cases} \tag{1}$$

In (1), the continuous-time dynamics (modeling a physical process) interacts with a discrete-time map (modeling the cyber component of the system and embedding the computational and communication/quantization elements of the CPS). In particular, (i) the continuous process (characterized by some state variable, $x$) gets its inputs, $u$, from an algorithm, $\mathscr{A}$; (ii) $e \in \mathscr{E}$ is the time dependent, possibly quantized, environmental variable taken as an input by the algorithm. The set $\mathscr{E} \subset \mathbb{R}^q$, $q \in \mathbb{N}$, $q \geq 1$ is a subset over which the environmental variable takes values. Note also that $\mathscr{A}$ takes as input (a possibly quantized version of) the state variable, $x$; and (iii) $\mathscr{U} := \{u_1, \ldots, u_q\}$ is the finite set of input vectors provided by the algorithm to the continuous process. In what follows, we denote by $\mathscr{S}_{in}$ the set of inputs to the algorithm and by $\mathscr{S}_{out}$ the set of outputs (see also Appendix A).

Throughout this paper, we assume that

**A1** $\mathscr{A}$ is correct;[23]

**A2** Let $\sigma(t) : [t_0, +\infty[ \to \mathscr{U}$ be a piecewise constant function. A unique forward complete Caratheodory solution[24] exists for the time dependent switching system $\dot{x} = f(t,x,\sigma)$.

## III. TECHNICAL LEMMAS

In this section, we present two technical results that will be later used to study networks of CPSs. We denote by $|v|$ a vector norm for the generic $n$-dimensional vector $v$ and by $\mu(A)$ the corresponding induced matrix measure of the $n \times n$ square matrix, $A$ (see Appendix A). The proofs, the definitions, and the mathematical background are given in Appendixes A and B.

**Lemma 1.** *Let A1 and A2 hold for (1). Additionally, assume that the following conditions are satisfied.*

- *For the algorithmic process*
  1. $(x,e) \in \mathscr{S}_{in}$, $\forall x \in \mathbb{R}^n$, *and* $\forall e \in \mathscr{E}$;
  2. *the set of vector inputs* $\mathscr{U}$ *is such that* $\mathscr{U} \subseteq \mathscr{S}_{out}$;
  3. *any input vector* $u \in \mathscr{U}$, *when set as an input to the continuous component of (1), remains so for at least some finite time.*
- *For the continuous process,*
  4. *the dynamics* $\dot{x} = f(t,x,u)$ *is always well-posed (see Definition 1 in Appendix A);*
  5. *there exists a matrix measure,* $\mu$, *such that*

$$\mu\left(\frac{\partial f}{\partial x}\right) \leq -c^2, \quad c \neq 0,$$

$\forall u \in \mathscr{U}, \forall x \in \mathbb{R}^n, \forall t \geq 0.$

*Then, for any two solutions of (1), say* $x_1(t)$ *and* $x_2(t)$, *it happens that*

$$|x_1 - x_2| \to 0, \quad t \to +\infty.$$

A CPS fulfilling the conditions of Lemma 1 is said to be CPS contracting and its trajectories globally converge toward each other. Lemma 1 extends to CPS the notion of contracting systems.[25,26]

With the next result, we give a sufficient condition for the convergence of the trajectories of the CPS (1) toward an $m$-dimensional ($m \leq n$) forward invariant subspace, $\mathscr{M}$.

**Lemma 2.** *Let A1 and A2 hold for (1). Additionally, assume that*

- *for the algorithmic process*
  1. $\forall x \in \mathbb{R}^n$ *and* $\forall e \in \mathscr{E}$, $(x,e) \in \mathscr{S}_{in}$;
  2. *the set of inputs* $\mathscr{U}$ *is such that* $\mathscr{U} \subseteq \mathscr{S}_{out}$;
  3. *any input vector* $u \in \mathscr{U}$, *when set as an input to the continuous component of (1), remains so for at least some finite time;*
- *the following conditions are satisfied for the continuous process*
  4. *there exists a subspace,* $\mathscr{M}$, *which is invariant for* $\dot{x} = f(t,x,u)$ *for any* $u \in \mathscr{U}$;
  5. *the dynamics* $\dot{x} = f(t,x,u)$ *is always well-posed (see Definition 1 in Appendix A);*
  6. *If we say* $V$ *the matrix spanning* $\mathscr{M}^{\perp}$, *there exists a matrix measure,* $\mu$, *such that*

$$\mu\left(V \frac{\partial f}{\partial x} V^T\right) \leq -c^2,$$

*with* $c \neq 0$, $\forall u \in \mathscr{U}$, $\forall x \in \mathbb{R}^n$, $\forall t \geq 0$.

*Then, all the trajectories of (1) converge toward* $\mathscr{M}$. *That is, for any solution* $x(t)$, *it holds that*

$$|Vx| \to 0, \quad t \to +\infty.$$

A CPS fulfilling the conditions of Lemma 2 is said to be CPS contracting relative to $\mathscr{M}$.

We wish to emphasize that the hypothesis of Lemmas 1 and 2 can be turned into design conditions when the goal is to design a contracting CPS fulfilling convergence between any two of its trajectories or relative to some invariant subspace, respectively.

**Remark 1.** *Lemmas 1 and 2 give two sufficient conditions to assess the contracting properties of CPSs of the form of (1). We will next show how these results can be used to study networks of CPSs with both directed and undirected topologies.*

### A. Example: Consensus with stubborn nodes

As a first example to illustrate our results, we now revisit the consensus problem,[8] but assume some of the nodes are "stubborn." That is, such nodes do not follow the normal update rule and their state is kept constant over time. This type of misbehaving behavior[28] is of interest in the context of social networks, where the stubborn nodes might represent agents that willingly do not change their opinion over time to, e.g., influence the outcome of a decision process.[29] In this example, we consider the network of Fig. 1 and assume that node 2 is stubborn, i.e., its state is kept constant, say equal to $\bar{x}_2$, regardless of the states of its neighbors. All the other network nodes will be termed as healthy in what follows. The network dynamics can be given as

$$\dot{x}_i = \sum_{j \in \mathcal{N}_i} (a_{ij} x_j - a_{ii} x_i)$$

for all healthy nodes, while $\dot{x}_2 = 0, x_2(0) = x_{20}$. Here, $\mathcal{N}_i$ denotes the set of neighbors to node $i$, while $a_{ij}$ are coefficients describing the interconnection graph and its weights. Clearly, consensus, if possible, can only be achieved if all healthy nodes converge onto the state of node 2.

We now seek to find a simple mechanism that allows the healthy network nodes to exclude the stubborn node and to achieve consensus among themselves. To this aim, in order to embed this mechanism in the network, we design a local algorithm, $\mathscr{A}_i$, interacting with the consensus dynamics at the nodes. This results in the CPS,

$$\begin{cases} \dot{x}_i = \sum_{j \in \mathcal{N}_i} (a_{ij} x_j - a_{ii} x_i), \\ (a_{ii}, a_{ij}) \leftarrow \mathscr{A}_i \left(x_i, x_j\right), \qquad j \in \mathcal{N}_i. \end{cases} \tag{2}$$

In (2), the coefficients $a_{ii}$'s and $a_{ij}$'s that regulate the consensus dynamics are generated by an algorithmic process, $\mathscr{A}_i$. In turn, this process is deployed on the $i$th node and takes as input the state of the $i$th node and of its neighbors.

We now show how Lemma 2 can be used to give guidelines on how to design a simple mechanism that allows healthy nodes of the network in Fig. 1 to (i) isolate the stubborn node and (ii) achieve consensus. A simple mechanism to achieve this goal is given in Algorithm 1. Such a mechanism has been obtained from the application of Lemma 2, which led to the following conceptual design steps.

**Step 1:** Ensure that each algorithm $\mathscr{A}_i$ can take as input the state vector (Hypothesis 1);

**Step 2:** Ensure that the continuous process can take as input $u_i$, generated by the algorithm and that the resulting dynamics for the continuous process is well-posed (Hypotheses 2 and 5);

**Step 3:** Constrain $u_i$ so that it does not switch arbitrarily fast (Hypothesis 3);

**Step 4:** Verify that the choice of parameters imposed by the algorithm ensures CPS contraction of the network dynamics toward the invariant subspace $\mathscr{M} := \{x_1 = x_3 = x_4 = x_5\}$ (Hypotheses 5 and 6). This is indeed true and can be verified by picking

$$V := \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

and by noticing that condition 6 of Lemma 2 is fulfilled.

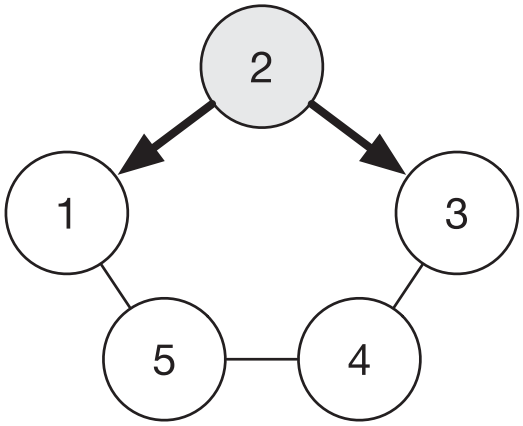


**FIG. 1.** The attack model considered in Sec. III A. Node 2 transmits an arbitrary, constant, value to its neighbors (node 1 and node 3), thus resulting in monodirectional connection from node 2 to nodes 1 and 3.

**Algorithm 1.** Detection for the attack model

```
function (a_ii, a_ij) = MECHANISM (x_i, STATE OF NEIGHBORS)
  while True do
    GET state from nearby nodes
                ▷ Check if neighbors changed their state:
    for j ∈ N_i do
      if x_j == x_{j,old} AND x_i ≠ x_j then
        a_ij ← 0
      else
        a_ij ← 1
      end if
    end for
    a_ii ← Σ_j a_ij
                ▷ Save the previous state of the neighbors:
    for j ∈ N_i do
      x_{j,old} ← x_j
    end for
  end while
end function
```

In Fig. 2, the time evolution for the network nodes is shown when the algorithm $\mathscr{A}_i$ is deployed onto the healthy nodes of the network. As shown in such a figure, the nodes are able to detect that node 2 is stubborn and are able to isolate it. Once this node is isolated, the healthy nodes achieve a consensus.

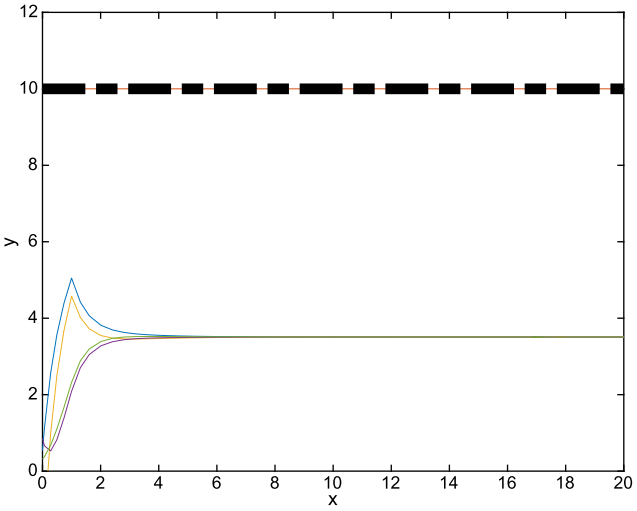


**FIG. 2.** Time evolution for the state variables, $x_i(t)$, of the network of Fig. 1. The value that node 2 transmits is $\bar{x}_2 = 10$. The figure shows that, thanks to the deployment of $\mathscr{A}$, healthy nodes are able to achieve consensus and disconnect from node 2.

### B. Example: Computing the weights of feed-forward neural networks to ensure its stability

In order to further illustrate our results, we now turn our attention to the problem of determining the weights of the neural network[30,31] of Fig. 3 in order to ensure its stability.Network topologies like the one in Fig. 3 naturally arise as the quotient network of directed feed-forward neural networks,[32] and, in this context, each node describes the dynamics of a network layer. We let $N$ be the number of nodes/layers and $x_i \in \mathbb{R}^n$ be the state variable of the $i$th node. The intrinsic dynamics of the $i$th node is modeled via the smooth function $f_i(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$. Moreover, the smooth function $h_{i,i}(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ models a self-regulation loop at node $i$, and $g_{i,i-1}(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ models the smooth coupling function between layer $i$ and layer $i-1$. Also, we let (i) $w_{i,i} \in \mathbb{R}$ be the weight for the self-regulation loop at node $i$ and (ii) $w_{i,i-1} \in \mathbb{R}$ be the coupling weight between layer $i$ and layer $i-1$. The vector of coupling weights, i.e., $\bar{w} = [w_{1,1}, w_{2,2}, w_{2,1}, \ldots, w_{N,N}, w_{N,N-1}]$ is computed by an algorithm, say $\mathscr{A}$, which takes as input the (known) functions $f_i$'s, $h_i$'s, and $g_{i,i-1}$'s. The goal of the algorithm is that of computing $\bar{w}$ to ensure stability of the network. The overall network dynamics can

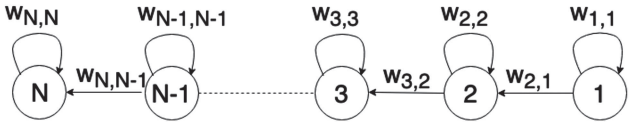


**FIG. 3.** Network topology for the example of Sec. III B. Each node can be used to model a layer of a feed-forward neural network, and the parameters $w_{ij}$'s denote its weights.

then be modeled via the CPS of the form

$$\begin{aligned} \dot{x}_1 &= f_1(x_1) + w_{1,1}h_1(x_1), \\ \dot{x}_i &= f_i(x_i) + w_{i,i}h_i(x_i) + w_{i,i-1}g_{i,i-1}(x_{i-1}), \quad i = 2, \ldots, N, \\ \bar{w} &\leftarrow \mathscr{A}(f_1, \ldots, f_N, h_1, \ldots, h_N, g_{2,1}, \ldots, g_{N,N-1}). \end{aligned} \tag{3}$$

Given this setup, a simple algorithmic mechanism to determine a set of weights making the network stable is given in Algorithm 2. This algorithm, which can be run whenever new layers are added to the network and/or the coupling/self-regulation functions change, has been obtained by applying Lemma 1 in order to guarantee CPS contraction of (3). It is indeed straightforward to verify that Algorithm 2 fulfills all the conditions of Lemma 1. In particular, let $J$ be the Jacobian of the continuous-time dynamics in (3). Then, condition 5 of Lemma 1 is fulfilled by Algorithm 2 by guaranteeing that $\mu_\infty(J)$ is uniformly negative definite [see Appendix A for the definition of the matrix measure $\mu_\infty(\cdot)$]. Finally, we remark here that the goal of Algorithm 2 is to give an algorithmic procedure to find the weights of the neural network in Fig. 3 so that it is stable. Hence, it can be used in conjunction with other algorithmic procedures, such as backpropagation, to solve inverse dynamics problems.[33]

**Algorithm 2.** Computing network weights ensuring CPS contraction

**function** $\bar{w}$ = WEIGHTS $(f_1, \ldots, f_N, h_1, \ldots, h_N, g_{2,1}, \ldots, g_{N,N-1})$
  GET the functions describing the intrinsic node dynamics and couplings, i.e., $f_1, \ldots, f_N, h_1, \ldots, h_N, g_{2,1}, \ldots, g_{N,N-1}$
  **for** $i = 1, \ldots N$ **do**
    $J_{ii} \leftarrow \max_{x_i} \mu_\infty \left( \frac{\partial f_i}{\partial x_i} + w_{i,i} \frac{\partial h_i}{\partial x_i} \right)$
    Tune $w_{i,i}$ so that $J_{ii} < 0$
    **if** $J_{i,i} < 0$ **then**
      $J_{i,i-1} \leftarrow \max_{x_j} \left\| \frac{\partial g_{i,i-1}}{\partial x_j} \right\|_\infty$
      Tune $w_{i,i-1}$ so that $\left| w_{i,i-1} \right| < -J_{i,i}/J_{i,i-1}$
    **end if**
  **end for**
**end function**

## IV. APPLICATIONS

We now consider two sample applications and illustrate how the methodology introduced in this paper can be used.

### A. Cooperative load management

The first application we present is primarily motivated by load balancing problems in the context of decentralized water management and smart/micro grids.[34,35] In our setup, we consider a network of interconnected "reservoirs" (e.g., water tanks for water networks

or batteries for smart grids). Each reservoir consists of a continuous process (i.e., the dynamics for the level of water in the reservoir or the level of energy in the battery) and of an algorithm responsible to set the policy of usage. We use Lemma 1 to design a decentralized cooperative load management system allowing each reservoir to handle its demand by collaborating with its neighbors. Here, the focus of this section is not on proposing novel solutions for this problem but rather to demonstrate the effectiveness of our approach and how this can be used to give design guidelines.

The mathematical model we consider in this section is

$$\begin{cases} \dot{x}_i = p_i(t) - d_i(t) + a_{ii} + \sum_{j \in \mathcal{N}_i} a_{ij}, & x_i(t_0) \geq 0, \\ (a_{ii}, a_{ij}) \leftarrow \mathcal{A}_i(x_i, x_j), & j \in \mathcal{N}_i, \end{cases} \tag{4}$$

where

- $x_i \in \mathbb{R}$ is the state variable of the $i$th network node ($i = 1, \ldots, N$). Physically, in a water management system, this variable denotes the level of water in the $i$th tank, while for electric batteries, this represents the $i$th battery level;
- $\mathcal{N}_i$ is the set of neighbors for the $i$th network node, i.e., the nodes with which the $i$th algorithm, $\mathcal{A}_i$, can collaborate;
- $d_i(t)$ models the exogenous demand requested to node $i$. Physically, this represents the request of water to the $i$th tank in a water management system or an energy request to the $i$th electric battery;
- $p_i(t)$ models an external source that node $i$ can use to satisfy the demand. From the physical viewpoint, in a water management system, this function models the water intake from a water source, while in an electric battery, this models the intake of energy from renewable sources;
- $(a_{ii}, a_{ij})$ is the control input to node $i$ returned by the decentralized control algorithm $\mathcal{A}_i$ within a finite time. As better detailed below, the control inputs physically are self-productions and node-to-node exchanges of water/energy.

The control action of the $i$th algorithm consists of two components, $a_{ii}$ and $a_{ij}$:

- $a_{ii}$ has the form $K_{ii}(t, x_i, x_j) - x_i$. Physically, $K_{ii}(\cdot)$ is a self production term for the $i$th reservoir. Note that this self-production depends on the state of the $i$th reservoir and of its neighbors. That is, the rate at which each reservoir self produces depends on the state of the node and on the state of its neighbors;
- $a_{ij}$ has the form $K_{ij}(t, x_i, x_j)$. Physically, such a term is an exchange term between node $i$ and node $j$. Also, in this case, the exchange depends on the state of the $i$th reservoir and of its neighbors.

The decentralized algorithm $\mathcal{A}_i$ presented here allows nodes to communicate and to coordinate their actions so that (i) each node is able to handle its demand, $d_i(t)$ and (ii) the excess between the production $p_i$ and the demand $d_i$ is shared with neighbors in need. The key idea for the decentralized algorithm $\mathcal{A}_i$ is that each node labels itself as a "Generator," "Consumer," or "Neutral." This classification depends on the balance between $p_i(t)$ and $d_i(t)$ and is determined by the algorithm $\mathcal{A}_i$, which has the following macrosteps:

**Algorithm 3.** Pseudocode for the load management algorithm

```
function (a_ii, a_ij) = LOADMANAGEMENT (p_i, d_i,
    STATE OF NEIGHBORS)
  while True do
    GET state from nearby nodes
    if p_i − d_i > 0 then
      LABEL ← "Generator"
    else if p_i − d_i < 0 then
      LABEL ← "Consumer"
    else p_i − d_i == 0
      LABEL ← "Neutral"
    end if
    if LABEL == "Generator" then
      K_ii ← 0
      AVAILABLE ← p_i − d_i
      GET requests from nearby Consumers
      CHECK feasibility of the requests
      SET K_ij < 0, ∑|K_ij| ≤ AVAILABLE, K_ij = −K_ji
    else if LABEL == "Neutral" then
      K_ii ← 0
      K_ij ← 0
      AVAILABLE ← 0
    else if LABEL == "Consumer" then
      FIND nearby generators
      K_ii ← −p_i + d_i
      while nearby generators do
        GET a fraction of AVAILABLE from the jth
            nearby generator (a_j)
        SET K_ij and K_ji (K_ij = a_j, K_ji = −K_ij)
        if p_i − d_i + ∑a_j < 0 then
          K_ii ← −p_i + d_i − ∑a_j
        else K_ii ← 0
        end if
      end while
    end if
    a_ii ← K_ii − x_i
    a_ij ← K_ij
  end while
end function
```

1. $\mathcal{A}_i$ gets data from nearby nodes. Specifically, the algorithm needs to know whether its neighbors are generators, consumers, or neutrals;

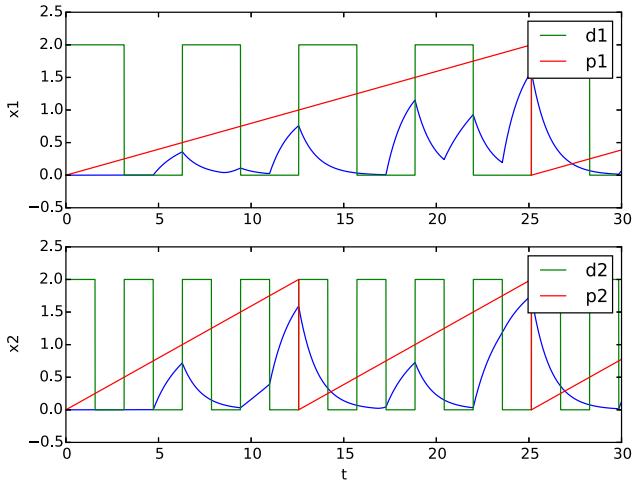


**FIG. 4.** Time evolution of the network state variables ($x_i$'s), together with the $d_i$'s and the $p_i$'s used in the simulation.

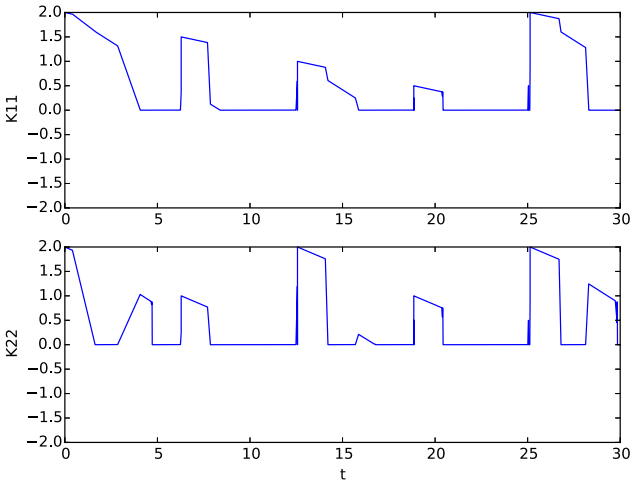


**FIG. 5.** Time evolution of the network nodes' self-production, $K_{11}$ and $K_{22}$.

2. Then, $\mathscr{A}_i$ determines whether its node is a generator ($p_i - d_i > 0$), a consumer ($p_i - d_i < 0$), or a neutral ($p_i - d_i = 0$);
3. Based on this, the following three behaviors are possible:
   - The algorithm behaves as a generator. In this case, it does not ask any support to nearby nodes nor it starts self-production ($K_{ii}$ and $K_{ij}$ are set to 0). The algorithm declares its availability to share part of its intake in excess ($p_i - d_i$). Then, the algorithm checks the requests from nearby consumers (if any) and shares with them part of its $p_i - d_i$ in excess;
   - The algorithm behaves as a neutral. In this case, it does not get energy from neighbors nor it produces by its own (i.e., $K_{ii}$ and $K_{ij}$ are set to 0);
   - The algorithm behaves as a consumer. In this case, it checks for nearby generators. If some of the neighbors is a generator, then the algorithm gets a portion of the intake in excess from each nearby generator (say $a_j$). Self-production occurs if the contribution from nearby generators is not enough, i.e., if $p_i - d_i + \sum_{j\in\mathscr{N}_i} a_j < 0$. In such a case, $K_{ii}$ is set so that $K_{ii} + p_i - d_i + \sum_{j\in\mathscr{N}_i} a_j \geq 0$.

The pseudocode for the decentralized algorithm is given in Algorithm 3 and proofs are given in Appendix B. In order to show the key features of the algorithm, we consider as a representative example a distributed CPS of 2 nodes. Figure 4 shows the time behavior of the nodes' state variables, together with their related exogenous demand and production functions. Such a figure shows that both $x_1$ and $x_2$ are always positive. That is, the algorithm allows each node to handle its demand (if a network node were not able to fulfill its demand, its state variable would become negative). The time evolution of $K_{11}$, $K_{12}$, $K_{22}$, and $K_{21}$ is shown in Figs. 5 and 6. Note that, in Fig. 6, a negative value for $K_{ij}$ means that node $i$ is giving to node $j$ part of its intake in excess so that it can contribute to satisfy the demand for node $j$. Finally, we wish to remark that, with Algorithm 3, nodes that label themselves as "Consumers" can only charge from their neighbors (if these are "Generators"). An interesting extension of such an algorithm, which will be subject of our future research, is that allowing Consumer nodes to collaborate not only with their neighbors but also with their neighbors' neighbors.

## B. Distributed queuing system

This section is motivated by scenarios where a number of users buys "a ticket" (i.e., each user reserves the right) to access a given and a shared service. Then, the system we are designing here has the goal of prioritizing groups of users by regulating the timing when they get access to the shared service. In order to better illustrate this motivating scenario, we consider users buying a ticket to fly from a given

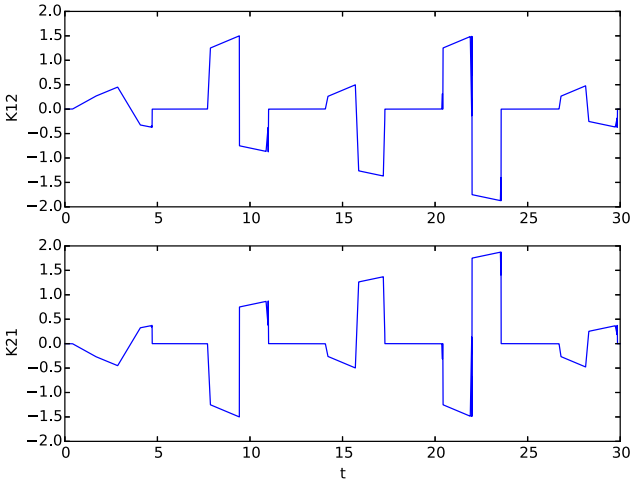


**FIG. 6.** Exchange between nodes.

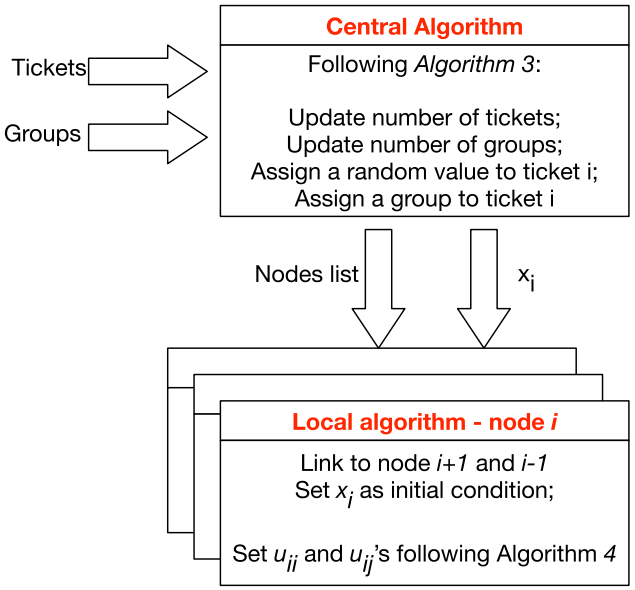


**FIG. 7.** Concept macroarchitecture and information exchange between system modules.

airport at a given time and where all the users need to do their check-in procedures (the use of check-in facilities is the shared service). In this context, the system considered in this section would assign different arrival times to passengers in order to reduce their waiting time at the airport. Our setup is schematically shown in Fig. 7; namely,the system consists of a central algorithm, which is responsible to collect the ticket requests from users and assign each user to a group. The central algorithm takes as input the number of groups, $N_G$, into which the users will be partitioned and the time window,

**Algorithm 4.** Pseudocode for the central algorithm

**function** $(X_0, G_X)$ = CENTRAL (Ticket, $T_{\max}$, $N_G$)
  **while** New Ticket OR Operator Command **do**
    UPDATE total number of tickets, $T$
    CREATE the $T$-dimensional vectors $X_0$, $G_X$
    INITIALIZE $V_G = [1, \ldots, N_G]$
    ▷ Set up the parameters for the decentralized algorithm:
    **for** $i$ in range(1,T) **do**
      $X_0(i) \leftarrow$ random number from (0, $T_{\max}$)
      $G_X(i) \leftarrow$ random value from ($V_G$)
    **end for**
  **end while**
**end function**

**Algorithm 5.** Pseudocode for the local algorithm

**function**($u_{ii}$, $u_{i,i+1}$, $u_{i,i-1}$) = LOCAL ($X_0(i)$, $G_X(i)$, $T$, $x_i$, $x_{i-1}$, $x_{i+1}$)
  **while** Input from Central **do**
    GET the group of node $i+1$: $G_X(i+1)$
    GET the group of node $i-1$: $G_X(i-1)$
    SET initial condition: $x_i(0) = X_0(i)$
    ▷ Groups creation:
    **if** $i \neq 1$ and $i \neq T$ **then**
      $u_{ii} \leftarrow -2$
      **if** $i$ and $i+1$ are in the same group **then**
        $u_{i,i+1} \leftarrow 1$
      **else if** $i$ and $i+1$ are in different groups **then**
        $u_{i,i+1} \leftarrow G_X(i)/G_X(i+1)$
      **end if**
      **if** $i$ and $i-1$ belong to the same group **then**
        $u_{i,i-1} \leftarrow 1$
      **else if** $i$ and $i+1$ are in different groups **then**
        $u_{i,i-1} \leftarrow G_X(i)/G_X(i-1)$
      **end if**
    **else if** $i == 1$ **then**
      $u_{ii} \leftarrow -1$
      $u_{i,i-1} \leftarrow 0$
      **if** $i$ and $i+1$ are in the same group **then**
        $u_{i,i+1} \leftarrow 1$
      **else if** $i$ and $i+1$ are in different groups **then**
        $u_{i,i+1} \leftarrow G_X(i)/G_X(i+1)$
      **end if**
    **else if** $i == T$ **then**
      $u_{ii} \leftarrow -1$
      $u_{i,i+1} \leftarrow 0$
      **if** $i$ and $i-1$ are in the same group **then**
        $u_{i,i-1} \leftarrow 1$
      **else if** $i$ and $i-1$ are in different groups **then**
        $u_{i,i-1} \leftarrow G_X(i)/G_X(i-1)$
      **end if**
    **end if**
    ▷ Set output:
    **if** $x_i$, $x_{i+1}$, $x_{i-1}$ are stable **then**
      BREAK
    **end if**
  **end while**
**end function**

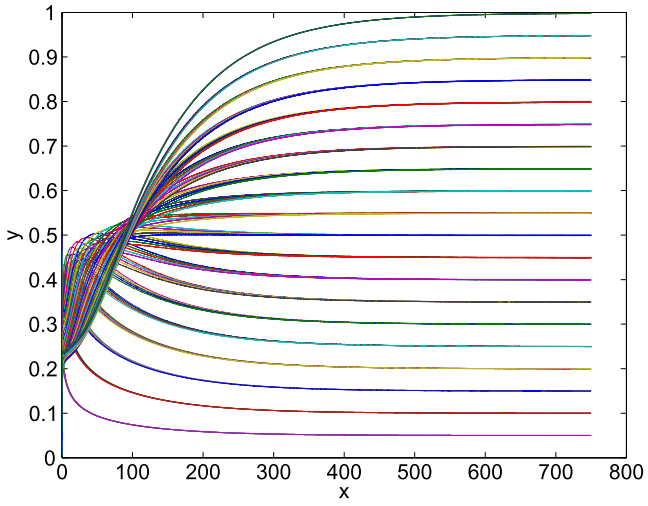


**FIG. 8.** Time evolution of $x_i$'s: groups converging toward an agreed arrival time. The arrival time for each user is the final value attained by $x_i$. Such a value is obtained in a distributed manner by applying the decentralized update law (5), designed following Lemma 2.

$T_{\max}$, in which users are spread. Based on such inputs, the algorithm generates a group membership for each user and assigns a random number to each user. The pseudocode for the central algorithm is given in Algorithm 4. Essentially, the goal of the central algorithm is to initialize a network that generates an arrival time for each user. Now, the key idea is to generate the arrival time for the $i$th user in a decentralized way, by means of a consensus algorithm. Specifically, let $x_i$ be the arrival time for the $i$th user. Then, this is generated following the update law,

$$\begin{cases} \dot{x}_i = u_{ii}x_i + u_{i,i+1}x_{i+1} + u_{i,i-1}x_{i-1}, \\ (u_{ii}, u_{i,i+1}, u_{i,i-1}) \leftarrow \mathscr{A}_i\,(x_i, x_{i+1}, x_{i-1})\,, \end{cases} \tag{5}$$

where $\mathscr{A}_i$ is a local algorithm, i.e., Algorithm 5. Such an algorithm, which has been designed by applying Lemma 2, is responsible to set the terms $u_{ii}$'s and $u_{ij}$'s so that (i) users belonging to the same group receive the same arrival time; (ii) each group has the same number of users; and (iii) the arrival times are uniformly spread across the allowable time window $T_{\max}$. Please see Appendixes A–C for further details.

As a representative example, we simulated our algorithm by considering a situation where 100 users buy a ticket for a given service and the operator sets the total number of groups to 20. Figure 8 shows that the decentralized system spontaneously prioritizes users and that the arrival times (the steady state values for $x_i$'s) for each group emerge as a collective behavior of the decentralized algorithm.

## V. CONCLUSIONS

In this paper, we considered the problem of characterizing the global emerging behavior in networks where nodes are modeled as CPSs. We first introduced two technical results to assess stability of such systems. The results give sufficient conditions for the stability of CPSs, where the physical component is a possibly nonlinear system, and it interacts with an algorithmic procedure. Then, we showed via a number of representative applications how our approach can be effectively used to design network CPSs that fulfill a given task of interest.

## ACKNOWLEDGMENTS

G. Russo was supported in part by a research grant from Science Foundation Ireland (SFI) under Grant No. 16/RC/3872.

## APPENDIX A: MATHEMATICAL BACKGROUND

Given a vector norm on Euclidean space $(|\cdot|)$, with its induced matrix norm $\|A\|$, the associated "matrix measure,"[36] $\mu$, is defined as $\mu(A) := \lim_{h\to 0^+} \frac{1}{h}\left(\|I + hA\| - 1\right)$. The matrix measures used in this paper can be easily computed and are reported in Table I.

### 1. Stability

We now give a survey[37] of the main results used in this paper. Consider the generic $n$-dimensional dynamical system

$$\dot{x} = f\,(t, x, \sigma)\,, \quad x(t_0) = x_0, \quad t_0 \geq 0, \tag{A1}$$

where $\sigma(t) : \mathbb{R}^+ \to \Sigma = \{\sigma_1, \ldots \sigma_p\}$ is a time dependent switching signal taking one over $p$ different values. Assume that Caratheodory solution exists for (A1) for all $t \geq t_0$. Caratheodory solutions are absolutely continuous functions satisfying (A1) almost everywhere in $t$.[24]

**Definition 1.** *We say that the dynamics $\dot{x} = f(t, x, \sigma)$ is "well posed" if the following technical conditions are fulfilled*: (*i*) *a Caratheodory solution exists and is forward complete*; (*ii*) *the map $(t, x) \mapsto f(t, x, \sigma)$ is continuous $\forall x \in \mathbb{R}^n$, $\forall \sigma \in \Sigma$, $\forall t \geq t_0$*; *and* (*iii*) *the map $x \mapsto f(t, x, \sigma)$ is continuously differentiable $\forall x \in \mathbb{R}^n$, $\forall \sigma \in \Sigma$, $\forall t \geq t_0$.*

The conditions above are standard conditions for systems of the form of (A1) and, intuitively, imply that the system of interest has a solution for all $t \geq t_0$ and that the distance between any two solutions does not exhibit discontinuities over time.

The main idea of contraction theory (and more generally of incremental stability methods) is to look at how the distance between any two solutions of (A1) evolves over time. Specifically, a system is said to be contracting if the distance between any two of its solutions (or trajectories) shrinks, i.e., the solutions "contract" toward

**TABLE I.** Useful matrix measures for the $n \times n$ dimensional matrix $A := [a_{ij}]$. The $i$th eigenvalue of $A$ is denoted with $\lambda_i(A)$.

| Vector norm, $\lvert\cdot\rvert$ | Induced matrix measure, $\mu\,(A)$ |
|---|---|
| $\lvert x\rvert_1 = \sum_{j=1}^{n} \lvert x_j\rvert$ | $\mu_1\,(A) = \max_j \left(a_{jj} + \sum_{i\neq j} \lvert a_{ij}\rvert\right)$ |
| $\lvert x\rvert_2 = \left(\sum_{j=1}^{n} \lvert x_j\rvert^2\right)^{\frac{1}{2}}$ | $\mu_2\,(A) = \max_i \left(\lambda_i \left\{\frac{A + A^T}{2}\right\}\right)$ |
| $\lvert x\rvert_\infty = \max_{1\leq j\leq n} \lvert x_j\rvert$ | $\mu_\infty\,(A) = \max_i \left(a_{ii} + \sum_{j\neq i} \mid a_{ij} \mid\right)$ |

each other. The following sufficient condition for the contraction of (A1) can be stated:[37]

**Theorem 1.** *Assume that for system (A1), the following conditions hold*: (1) *it is well posed*; (2) *there exists a matrix measure*, $\mu$, *and a constant* $c \neq 0$ *such that* $\mu\left(\frac{\partial f}{\partial x}(t,x,\sigma)\right) \leq -c^2$, $\forall t \geq 0$, $\forall \sigma \in \Sigma$, *and* $\forall x \in \mathbb{R}^n$. *Then, the distance between any two solutions of (A1) shrinks, i.e., for any two solutions* $x(t)$ *and* $y(t)$ *of (A1), it happens that* $|x(t)-y(t)| \leq |x(t_0)-y(t_0)|\, e^{-c^2(t-t_0)}$.

A system fulfilling the conditions of Theorem 1 is said to be contracting. Sometimes, in applications, we are not interested in proving that all trajectories converge toward each other. We are rather interested in proving that trajectories converge toward some subspace. Let $\mathscr{M}$ be an $m$-dimensional ($m \leq n$) invariant subspace for (A1), i.e., solutions of (A1) remain in $\mathscr{M}$ if their initial conditions are in $\mathscr{M}$ (this is formalized with the condition: $f(t,x,\sigma) = 0$, $\forall x \in \mathscr{M}$, $\forall t$, and $\forall \sigma$). Let (i) $w_i$ ($i = 1,\ldots,m$) be $n$-dimensional orthonormal vectors spanning $\mathscr{M}$, with $W^T = (w_1,\ldots,w_m) \in \mathbb{R}^{n\times m}$ and (ii) $v_i$ ($i = 1,\ldots,n-m$) be an orthonormal basis for $\mathscr{M}^{\perp}$, with $V^T = (v_1,\ldots,v_m) \in \mathbb{R}^{n\times n-m}$. Note that the vector $Vx$ belongs to $\mathscr{M}^{\perp}$, and, therefore, distance between $x$ and $\mathscr{M}$ is simply $|Vx|$. The following result holds, which provides a sufficient condition for the distance between $x$ and $\mathscr{M}$ to shrink.

**Theorem 2.** *Assume that for system (A1), the following conditions hold*: (1) *it is well posed*; (2) *for any* $\sigma \in \Sigma$, *there exists an invariant linear subspace* $\mathscr{M}$; (3) *there exists a matrix measure*, $\mu$, *and* $c \neq 0$ *such that* $\mu\left(V\frac{\partial f}{\partial x}(t,x,\sigma)\,V^T\right) \leq -c^2$, $\forall t \geq 0$, $\forall \sigma \in \Sigma$, *and* $\forall x \in \mathbb{R}^n$. *Then, any solution* $x(t)$ *of (A1) contracts toward* $\mathscr{M}$, *i.e., it happens that* $|Vx(t)| \leq |Vx(t_0)|\, e^{-c^2(t-t_0)}$.

A system fulfilling the conditions of Theorem 2 is said to be contracting relative to $\mathscr{M}$.[27] Note that if a system is contracting relative to a subspace, $\mathscr{M}$, then the distance between its solutions and $\mathscr{M}$ shrinks. In this sense, as discussed in the next example, the notion of contraction relative to a subspace is weaker than the notion of contraction, where we ask for all the distances between any solution of the system to shrink.

### 2. Algorithm correctness

In this section, we review some concepts related to algorithms relevant for this paper[23] for a detailed survey of these concepts. An algorithm, $\mathscr{A}$, transforms a set of input data into a set of output data. This transformation occurs through a set of procedural rules that allows one to perform a given "task." In turn, the task is defined through a set of specifications: (i) the name of the algorithm (typically, optional); (ii) initial conditions, i.e., the set of values/symbols on which the algorithm has to work; and (iii) final conditions, i.e., the set of values/symbols that the algorithm must return.

Given an input set, $\mathscr{S}_{in}$, we say that this is correct if it satisfies the initial conditions of the specifications. Analogously, we say that the output set, $\mathscr{S}_{out}$, is correct if this satisfies the final conditions of the specification. We are now ready to give the following definition:

**Definition 2.** *Algorithm* $\mathscr{A}$ *is said to be correct if for any input belonging to* $\mathscr{S}_{in}$ : (*i*) $\mathscr{A}$ *stops and* (*ii*) *the output belongs to* $\mathscr{S}_{out}$.

## APPENDIX B: PROOF OF THE TECHNICAL RESULTS

### 1. Proof of Lemma 1

We prove the result by considering the following CPS (1):

$$\begin{cases} \dot{y} = f(t,y,u), & y \in \mathbb{R}^n, \quad u \in \mathscr{U}, \\ u \leftarrow \mathscr{A}(x,e), & x \in \mathbb{R}^n, \quad e \in \mathscr{E}, \end{cases} \tag{B1}$$

and by noting that by construction,

- The solutions of (1) are particular solutions of system (B1). That is, system (B1) embeds the solutions of (1) and, therefore, it will be termed as "virtual system" in what follows;
- The input, $u$, is an exogenous time dependent switching input for the continuous part of the cyber-physical virtual system.

Note that, by hypothesis, assumption A1 is fulfilled, and, hence, $\mathscr{A}$ is correct. Moreover, hypothesis 1 guarantees that $\mathscr{A}$ always receives as input an element $(x,e)$, which belongs to $\mathscr{S}_{in}$. Therefore, for any element of $(x,e)$, the algorithm returns an output belonging to $\mathscr{S}_{out}$, which is a subset of $\mathscr{U}$ (hypothesis 2). Note also that, by hypothesis 3, $u(t)$ is a piecewise constant function, and, hence, by means of assumption A2, the dynamics,

$$\dot{y} = f(t,y,u(t)), \tag{B2}$$

admits a forward complete Caratheodory solution. Now, since $u(t)$ is an exogenous input to (B2), it follows that contraction of (B2) implies that all the solutions of (B1) also converge toward each other. The next step of the proof is then to show contraction of (B2). This can be immediately proved by noticing that, by means of hypotheses 4 and 5, all the hypotheses of Theorem 1 are fulfilled for (B2). Therefore, it can be shown[37] that for any two solutions of (B2), say $y_1(t)$ and $y_2(t)$, it happens that

$$|y_1(t)-y_2(t)| \leq |y_1(t_0)-y_2(t_0)|\, e^{-c^2(t-t_0)}.$$

This implies that all the solutions of (B1) converge toward each other.

Finally, recall that the solutions of (1) are also solutions of the virtual system. Therefore, let $x_1(t)$ and $x_2(t)$ be any two solutions of (1), and it happens that $|x_1(t)-x_2(t)| \to 0$, $t \to +\infty$, thus proving the result.

### 2. Proof of Lemma 2

The spirit of this proof is similar to the one of Lemma 1. Consider the following cyber-physical virtual system for system (1):

$$\begin{cases} \dot{y} = f(t,y,u), & y \in \mathbb{R}^n, \quad u \in \mathscr{U}, \\ u \leftarrow \mathscr{A}(x,e), & x \in \mathbb{R}^n, \quad e \in \mathscr{E}. \end{cases}$$

As for the proof of Lemma 1, we can prove this result by showing that the solutions of

$$\dot{y} = f(t,y,u)$$

converge toward $\mathscr{M}$. Since by hypothesis 4 the invariant subspace $\mathscr{M}$ exists for any choice of the input, $u$, we can consider the system

dynamics transversal to $\mathscr{M}$,

$$\begin{aligned} V\dot{y} &= Vf(t,y,u) \\ &= Vf(t,(V^TV+W^TW)y,u). \end{aligned}$$

Let $z := Vy$, we finally have

$$\dot{z} = Vf(t, V^Tz + W^TWy, u). \tag{B3}$$

Now, note that by hypotheses:

- a unique forward complete Caratheodory solution exists;
- $Vf(t, W^TWy, u) = 0$ (this follows from the fact that, since $W$ spans $\mathscr{M}$ and $V$ spans $\mathscr{M}^T$, then $V^TV + W^TW = I$) and, therefore, $z \equiv 0$ is a solution of (B3).

Thus, contraction of (B3) implies that all of its (Caratheodory) solutions converge toward $z \equiv 0$. Now, by means of hypothesis 6, we know that (B3) is contracting, and, hence, it implies that

$$|z(t)| \to 0, \quad t \to +\infty.$$

Finally, since $|z(t)| = |Vy(t)|$, we have that $|Vy(t)| \to 0$ as $t \to +\infty$. Since the solutions of (1) are particular solutions of the virtual system, we have that $|Vx| \to 0$, thus proving the result.

## APPENDIX C: ANALYSIS FOR THE APPLICATIONS

### 1. Cooperative load management system

We now use Lemma 1 to show contraction of (4) when this interacts with the decentralized algorithm introduced in Sec. IV A. In order to do so, first note that hypotheses 1–4 of Lemma 1 are fulfilled by construction by the decentralized algorithm. The only condition that needs to be checked is, therefore, hypothesis 5 of Lemma 1. This can be immediately proved by noting that the dynamics of the continuous process can be written as $\dot{x}_i = p_i(t) - d_i(t) + K_{ii}(t) + \sum_{j\in\mathscr{N}_i} K_{ij}(t) - x_i(t)$, where $K_{ij}$'s and $K_{ii}$'s are piecewise constant functions generated by the decentralized algorithm $\mathscr{A}_i$. Note that the vector field of such dynamics is continuous in the state, and this, together with the fact that $K_{ij}$'s and $K_{ii}$'s are piecewise constant functions, implies that a Caratheodory solution exists[24] (i.e., Assumption A2 is fulfilled). Hence, contraction of (4) can be then proved by noticing that the system Jacobian for each of the nodes is scalar and equal to $-1$.

Now, the next step is to show that the decentralized control algorithm effectively fulfills its control task. Please note that the control problem is solved by the algorithm if it keeps all the network state variables non-negative, i.e., $x_i(t) \geq 0$, $\forall t$. Therefore, we need to show that the algorithm "forces" the existence of a solution, say $x_d(t) = [x_{d,1}, \ldots, x_{d,N}]^T$ such that $x_{d,i}(t) \geq 0$, $\forall t \geq t_0$, $i = 1, \ldots, N$. This can be proved by noticing that by construction, the algorithm forces the quantity $p_i(t) - d_i(t) + K_{ii} + \sum_j K_{ij}$ to be non-negative, implying that the system is forced to evolve on the positive orthant.

### 2. Queuing system

We now use the ideas of Sec. III to show that the CPS in (5) converges toward a state where nodes belonging to the same group attain the same value. In order to do so, note that the algorithmic procedure $\mathscr{A}_i$ in (5) fulfills hypotheses 1–6 of Lemma 2. Moreover, the algorithm transforms network dynamics in $\dot{x}_i = \sum_{j\in\mathscr{N}_i}(\gamma_{ij}(x_j) - \gamma_{ii}(x_i))$, where $\gamma$'s are the coupling functions between nodes. Let $G_A, G_B \in [0, \ldots, N_G]$ be two different groups assigned by the central algorithm. Then, $\mathscr{A}_i$ sets the coupling function as

$$\gamma_{ij}(x) = \begin{cases} x & \text{if } i,j \text{ belong to the same group,} \\ \frac{G_i}{G_j}x & \text{if } i \in G_i \text{ and } j \in G_j. \end{cases}$$

Now, consider the following transformation for the network state variables: $x_i^* = \frac{1}{G_i}x_i$, $i \in G_i$. Then, under this transformation, the network dynamics of the CPS (5) becomes

$$\dot{x}_i^* = \sum_{j\in\mathscr{N}_i}\left(x_j^* - x_i^*\right). \tag{C1}$$

Now, a solution exists for the above dynamics, and, as shown in Sec. III, such a dynamics fulfills the hypotheses of Lemma 2 with $\mathscr{M} := \left\{x^* : x_i^* = x_j^*\right\}$. That is, (C1) is CPS contracting relative to $\mathscr{M}$. This in turn implies that network nodes reach an agreement onto a common value, i.e., $x_i^* \to x^*$ for all $i$. The result is then proved by noticing that this condition implies that $x_i \to G_ix^*$ (one can show that the algorithm is exploiting symmetries in the network vector field[38] to ensure convergence to the desired agreement).

## REFERENCES


[1] E. Lee, "The past, present and future of cyber-physical systems: A focus on models," Sensors **15**, 4837 (2015).
[2] A. R. Al-Ali, R. Gupta, and A. A. Nabulsi, "Cyber physical systems role in manufacturing technologies," AIP Conf. Proc. **1957**, 050007 (2018).
[3] V. Muliukha, A. Ilyashenko, V. Zaborovsky, and A. Lukashin, "Cyber-physical approach to the network-centric robotics control task," AIP Conf. Proc. **1776**, 090056 (2016).
[4] S. H. Strogatz, "Exploring complex networks," Nature **410**, 268 (2001).
[5] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," Rev. Mod. Phys. **74**, 47–97 (2002).
[6] S. Boccaletti, R. Criado, M. Romance, and J. J. Torres, "Introduction to focus issue: Complex dynamics in networks, multilayered structures and systems," Chaos **26**, 065101 (2016).
[7] L. M. Pecora and T. L. Carroll, "Synchronization of chaotic systems," Chaos **25**, 097611 (2015).
[8] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," IEEE Trans. Automat. Contr. **49**, 1520–1533 (2004).
[9] C. Qian, J. Cao, J. Lu, and J. Kurths, "Adaptive bridge control strategy for opinion evolution on social networks," Chaos **21**, 025116 (2011).
[10] G. Russo and J. J. E. Slotine, "Global convergence of quorum-sensing networks," Phys. Rev. E **82**, 041919 (2010).
[11] J. M. V. Grzybowski, E. E. N. Macau, and T. Yoneyama, "On synchronization in power-grids modelled as networks of second-order Kuramoto oscillators," Chaos **26**, 113113 (2016).
[12] K. Xi, J. L. A. Dubbeldam, and H. X. Lin, "Synchronization of cyclic power grids: Equilibria and stability of the synchronous state," Chaos **27**, 013109 (2017).
[13] Y. Tang, F. Qian, H. Gao, and J. Kurths, "Synchronization in complex networks and its application—A survey of recent advances and challenges," Annu. Rev. Control **38**, 184–198 (2014).
[14] W. Lu, B. Liu, and T. Chen, "Cluster synchronization in networks of coupled nonidentical dynamical systems," Chaos **20**, 013120 (2010).
[15] G. Russo, "How to desynchronize quorum-sensing networks," Phys. Rev. E **95**, 042312 (2017).

[16]G. Russo, "Loss of coordination in complex directed networks: An incremental approach based on matrix measures," Int. J. Robust Nonlinear Control **28**, 120–131 (2018).
[17]G. Fortino, R. Gravina, W. Russo, and C. Savaglio, "Modeling and simulating internet-of-things systems: A hybrid agent-oriented approach," Comput. Sci. Eng. **19**, 68–76 (2017).
[18]D. Zhang, Z. Xu, H. R. Karimi, Q. Wang, and L. Yu, "Distributed $h_\infty$ output-feedback control for consensus of heterogeneous linear multiagent systems with aperiodic sampled-data communications," IEEE Trans. Ind. Electron. **65**, 4145–4155 (2018).
[19]D. Zhang, L. Liu, and G. Feng, "Consensus of heterogeneous linear multiagent systems subject to aperiodic sampled-data and DoS attack," IEEE Trans. Cybern. **49**, 1501–1511 (2019).
[20]J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, "Toward a science of cyber-physical system integration," Proc. IEEE **1000**, 29–44 (2012).
[21]C. Nowzari and J. Cortes, "Team-triggered coordination for real-time control of networked cyber-physical systems," IEEE Trans. Automat. Contr. **61**, 34–47 (2016).
[22]E. Lee and S. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*, 2nd ed. (MIT Press, 2017).
[23]T. H. Cormen, *Introduction to Algorithms* (MIT Press, 2009).
[24]J. Cortes, "Discontinuous dynamical systems," IEEE Control Syst. Mag. **44**, 36–73 (2005).
[25]W. Lohmiller and J. J. E. Slotine, "On contraction analysis for non-linear systems," Automatica **34**, 683–696 (1998).
[26]G. Russo, M. di Bernardo, and J. Slotine, "A graphical algorithm to prove contraction of nonlinear circuits and systems," IEEE Trans. Circuits Syst. I **58**, 336–348 (2011).
[27]Q. C. Pham and J. J. E. Slotine, "Stable concurrent synchronization in dynamic system networks," Neural Netw. **20**, 62–77 (2007).
[28]M. Mobilia, "Does a single zealot affect an infinite group of voters?," Phys. Rev. Lett. **91**, 028701 (2003).
[29]D. D. Chinellato, I. R. Epstein, D. Braha, Y. Bar-Yam, and M. A. M. de Aguiar, "Dynamical response of networks under external perturbations: Exact results," J. Stat. Phys. **159**, 221–230 (2015).
[30]H. Chen, G. He, F. Huang, C. Shen, and Z. Hou, "Explosive synchronization transitions in complex neural networks," Chaos **23**, 033124 (2013).
[31]X. Li, J. Zhang, and M. Small, "Self-organization of a neural network with heterogeneous neurons enhances coherence and stochastic resonance," Chaos **19**, 013126 (2009).
[32]M. A. D. Aguiar, A. P. S. Dias, and F. Ferreira, "Patterns of synchrony for feed-forward and auto-regulation feed-forward neural networks," Chaos **27**, 013103 (2017).
[33]P. Beim Graben and R. Potthast, "Inverse problems in dynamic cognitive modeling," Chaos **19**, 015103 (2009).
[34]S. Arianos, E. Bompard, A. Carbone, and F. Xue, "Power grid vulnerability: A complex network approach," Chaos **19**, 013119 (2009).
[35]A. Yazdani and P. Jeffrey, "Complex network analysis of water distribution systems," Chaos **21**, 016111 (2011).
[36]M. Vidyasagar, *Nonlinear Systems Analysis*, 2nd ed. (Prentice-Hall, Englewood Cliffs, NJ, 1993).
[37]M. di Bernardo, D. Liuzza, and G. Russo, "Contraction analysis for a class of non-differentiable systems with applications to stability and network synchronization," SIAM J. Control Optim. **52**, 3203–3227 (2014).
[38]D. Fiore, G. Russo, and M. di Bernardo, "Exploiting nodes symmetries to control synchronization and consensus patterns in multiagent systems," IEEE Control Syst. Lett. **1**, 364–369 (2017).